Problem 7 (30 Points)

Problem Description

In this problem, you are given a dataset with two input features and one output. You will use a regression tree to make predictions for this data, evaluating each model on both training and testing data. Then, you will repeat this for multiple random forests.

Fill out the notebook as instructed, making the requested plots and printing necessary values.

You are welcome to use any of the code provided in the lecture activities.

Summary of deliverables:

- RMSE function
- Create 4 decision tree prediction surface plots
- Create 4 random forest prediction surface plots
- Print RMSE for train and test data for 4 decision tree models
- Print RMSE for train and test data for 4 random forest models
- Answer the 3 questions posed throughout

Imports and Utility Functions:

In [2]:
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm

def make_plot(X,y,model, title=""):
    res = 100
    xrange = np.linspace(min(X[:,0]),max(X[:,0]),res)
    yrange = np.linspace(min(X[:,1]),max(X[:,1]),res)
    x1,x2 = np.meshgrid(xrange,yrange)
    xmesh = np.vstack([x1.flatten(),x2.flatten()]).T
    z = model.predict(xmesh).reshape(res,res)

    fig = plt.figure(figsize=(12,10))
    plt.subplots_adjust(left=0.3,right=0.9,bottom=.3,top=.9)
    ax = fig.add_subplot(111, projection='3d')
    ax.plot_surface(x1,x2,z,cmap=cm.coolwarm,linewidth=0,alpha=0.9)
    ax.scatter(X[:,0],X[:,1],y,'o',c='black')
    ax.set_xlabel('$x_1$')
    ax.set_ylabel('$x_2$')
    ax.set_zlabel('y')
    plt.title(title)
    plt.show()
```

Load the data

Use the `np.load()` function to load "w5-hw2-train.npy" (training data) and "w5-hw2-test.npy" (testing data). The first two columns of each are the input features. The last column is the output. You should end up with 4 variables, input and output for each of the datasets.

In [4]:
```
# YOUR CODE GOES HERE
train = np.load("data/w5-hw2-train.npy")
test = np.load("data/w5-hw2-test.npy")
train_x = train[:,:2]
train_y = train[:,2]
test_x = test[:,:2]
test_y = test[:,2]
```

RMSE function

Complete a root-mean-squared-error function, `RMSE(y, pred)`, which takes in two arrays, and computes the RMSE between them:

In [23]:
```
def RMSE(y, pred):
    # YOUR CODE GOES HERE
    return np.sqrt(np.mean((pred-y)**2))
```

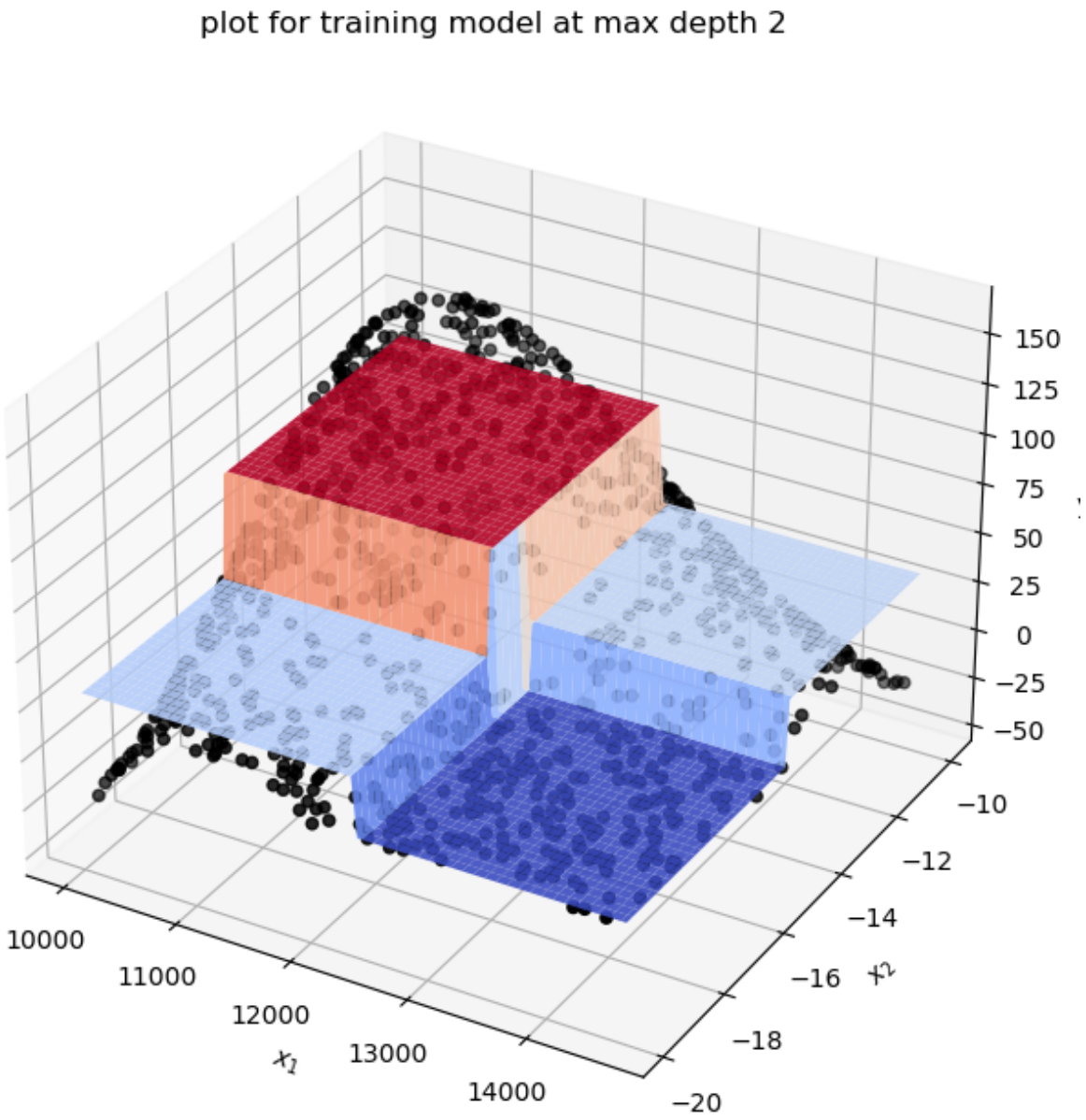
Regression trees

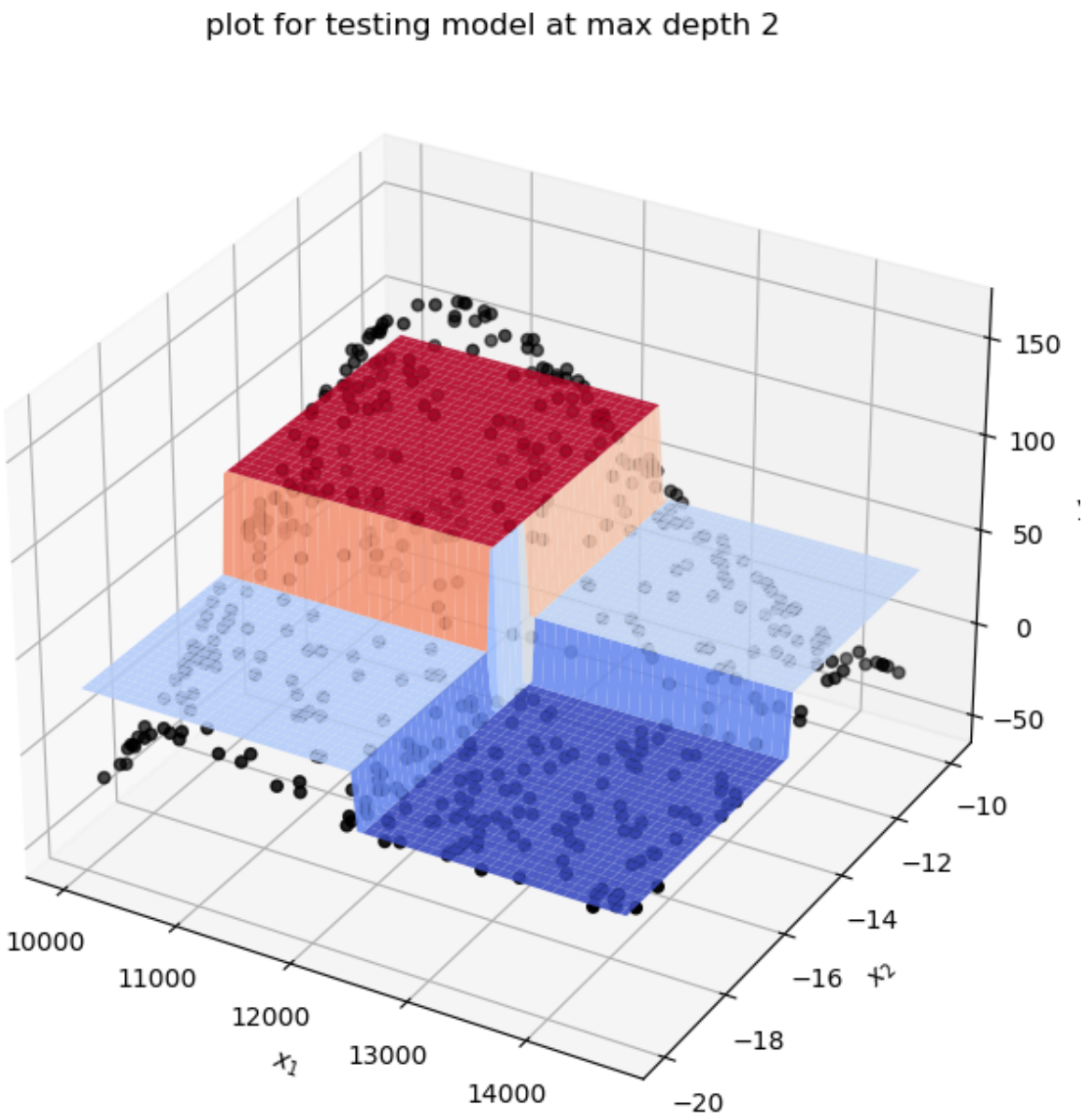
Train 4 regression trees in sklearn, with max depth values [2,5,10,25]. Train your models on the training data.

Plot the predictions as a surface plot along with test points -- you can use the provided function: `make_plot(X, y, model, title)`.
For each model, compute the train and test RMSE by calling your RMSE function. Print these results.

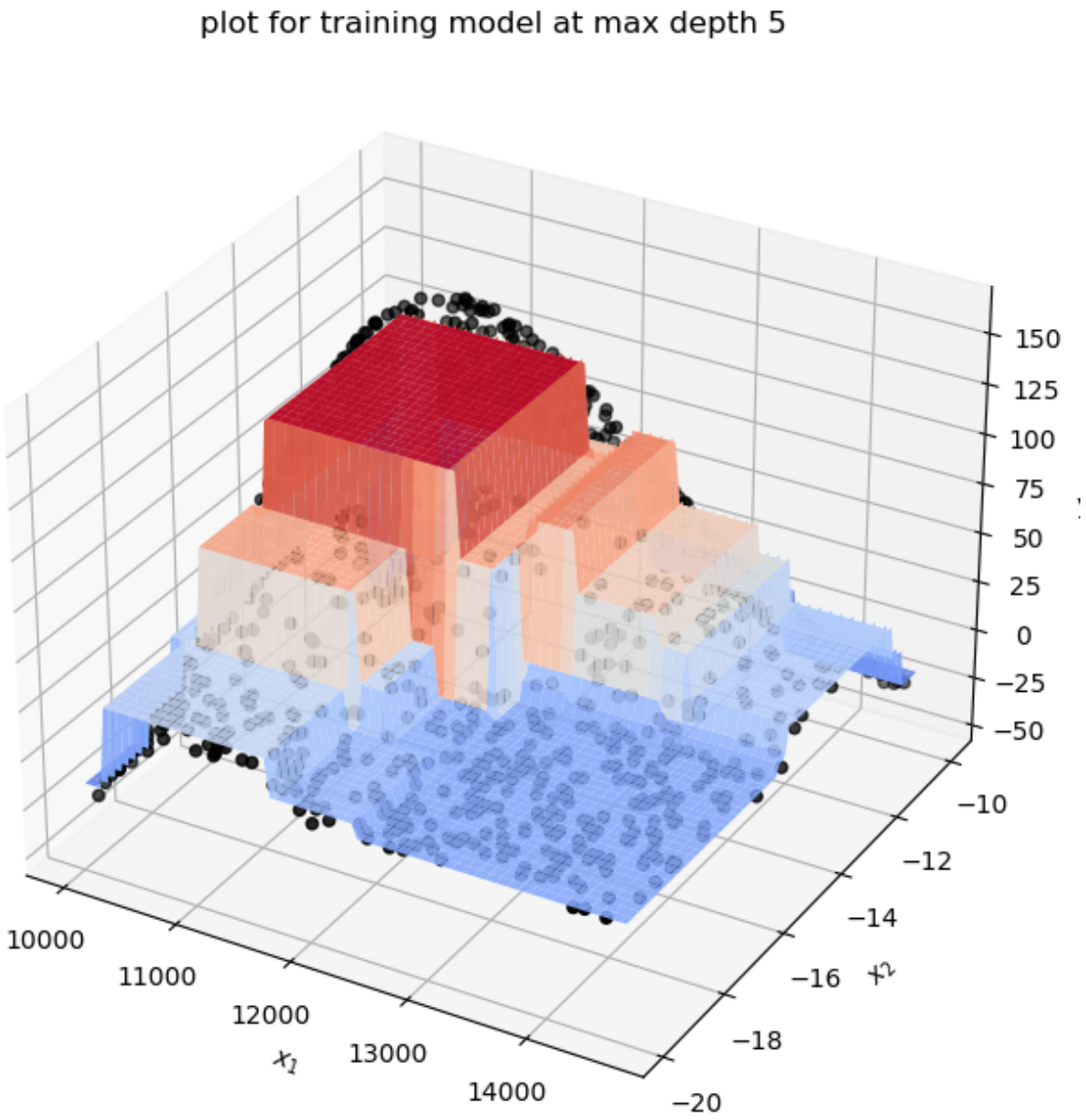
In [29]:
```
# YOUR CODE GOES HERE
for max_depth in [2,5,10,25]:
    # YOUR CODE GOES HERE
    # Create and fit `dt`
    dt = DecisionTreeRegressor(max_depth = max_depth,random_state = 0)
    dt.fit(train_x,train_y)
    pred_train = dt.predict(train_x)
    print("RMSE for train data: ", RMSE(train_y,pred_train))
    pred_test = dt.predict(test_x)
    print("RMSE for test data: ", RMSE(test_y,pred_test))
    make_plot(train_x,train_y,dt,f'plot for training model at max depth {max_depth}')
    make_plot(test_x,test_y,dt,f'plot for testing model at max depth {max_depth}')
```

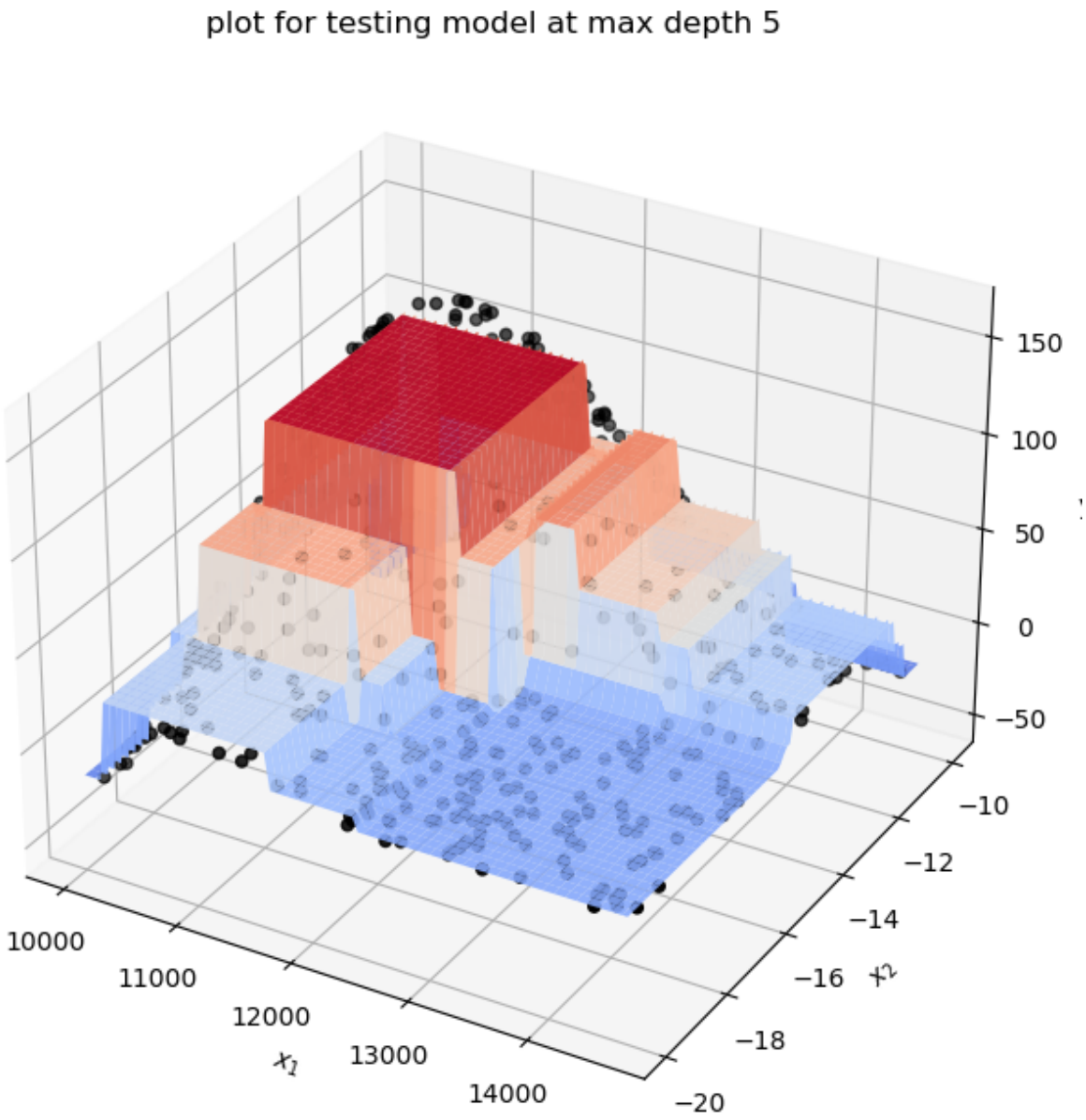
```
RMSE for train data:  35.47184989095342
RMSE for test data:  37.54886839401237
```


plot for training model at max depth 2
150
125
100
75
50
25
0
−25
−50
10000
11000
12000
13000
14000
x_1
−10
−12
−14
−16
−18
−20
x_2

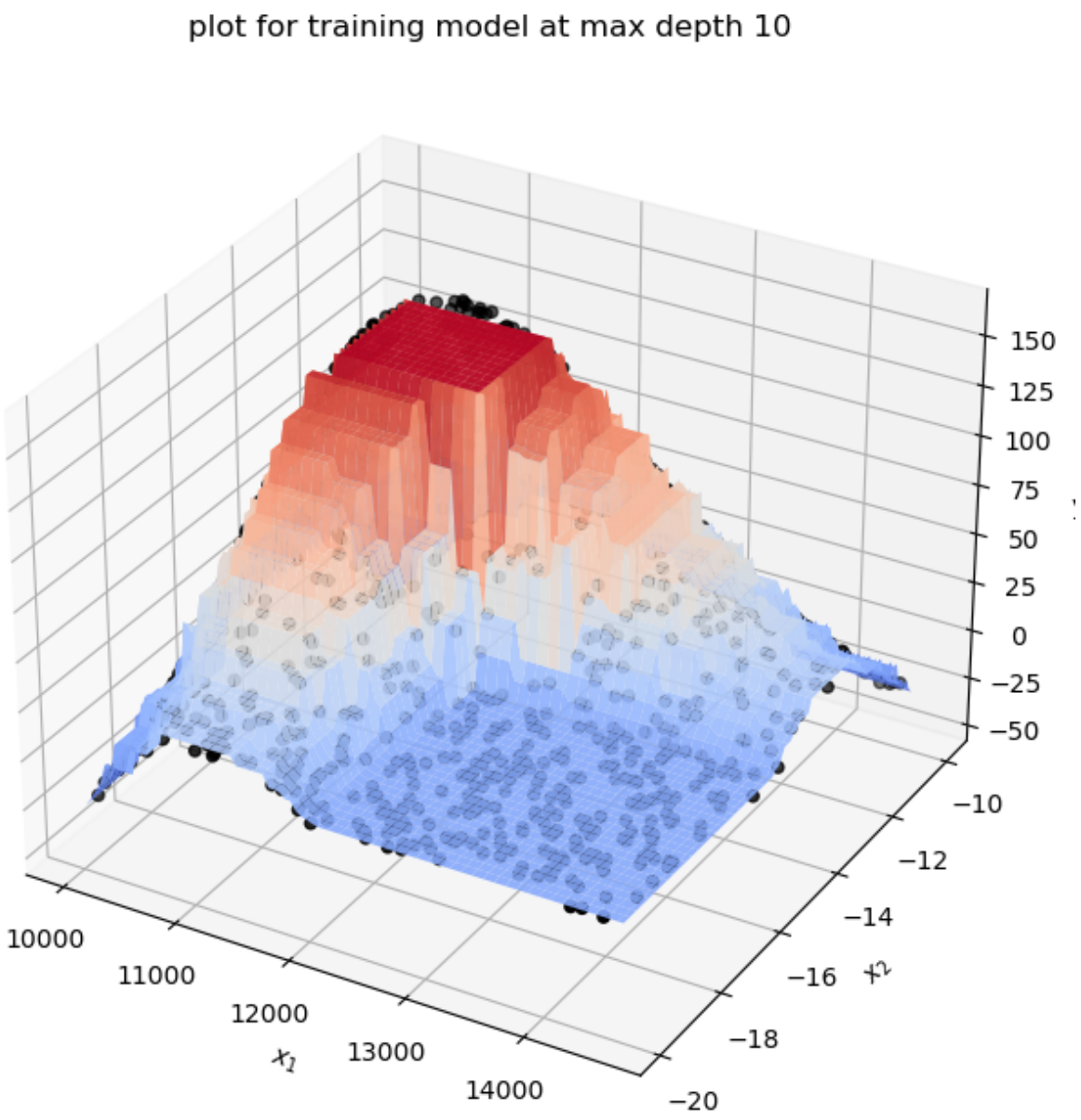


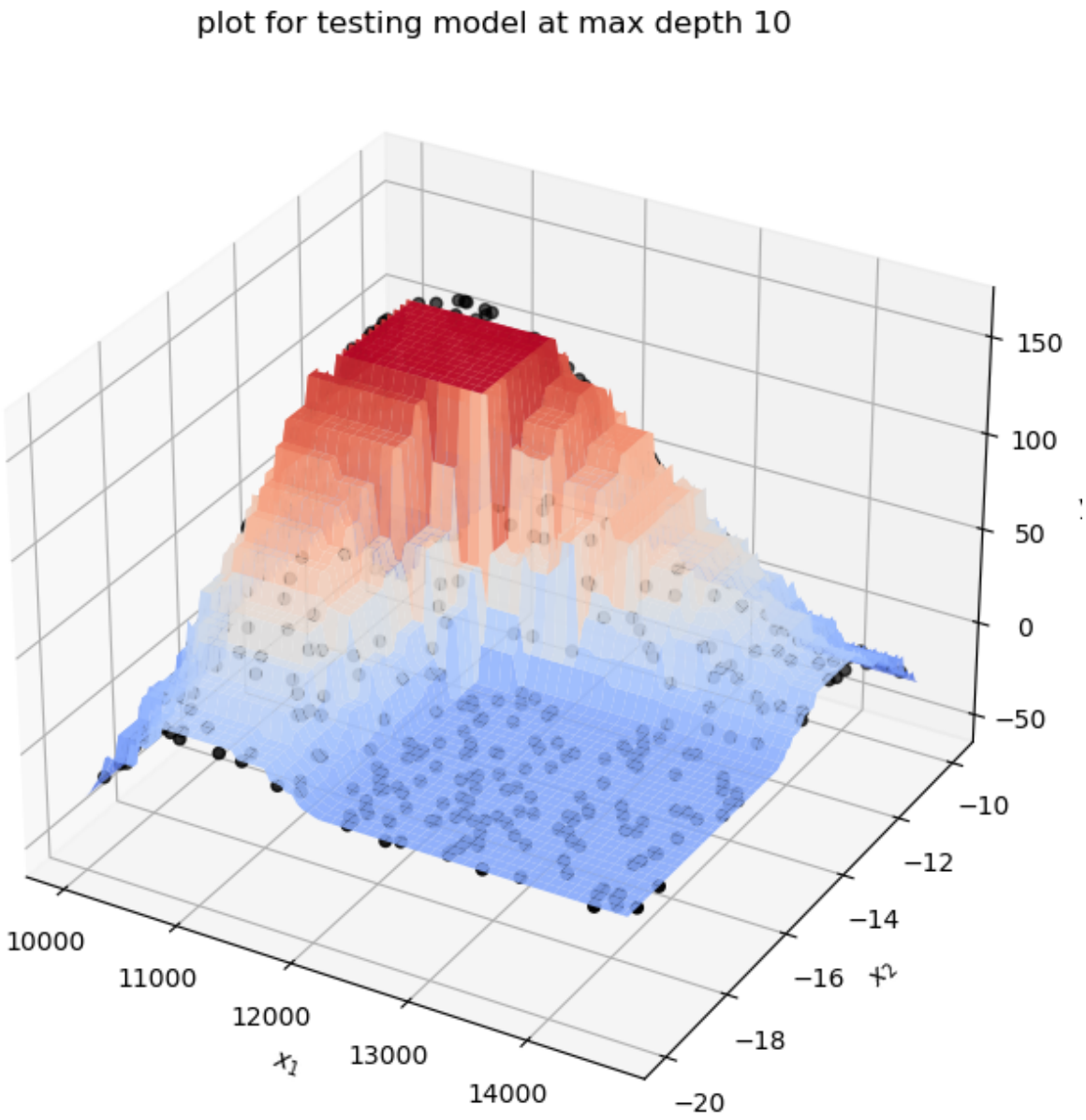

```
RMSE for train data:  17.932673237502154
RMSE for test data:  19.02935744931633
```


plot for training model at max depth 5
150
125
100
75
50
25
0
−25
−50
10000
11000
12000
13000
14000
x_1
−10
−12
−14
−16
−18
−20
x_2

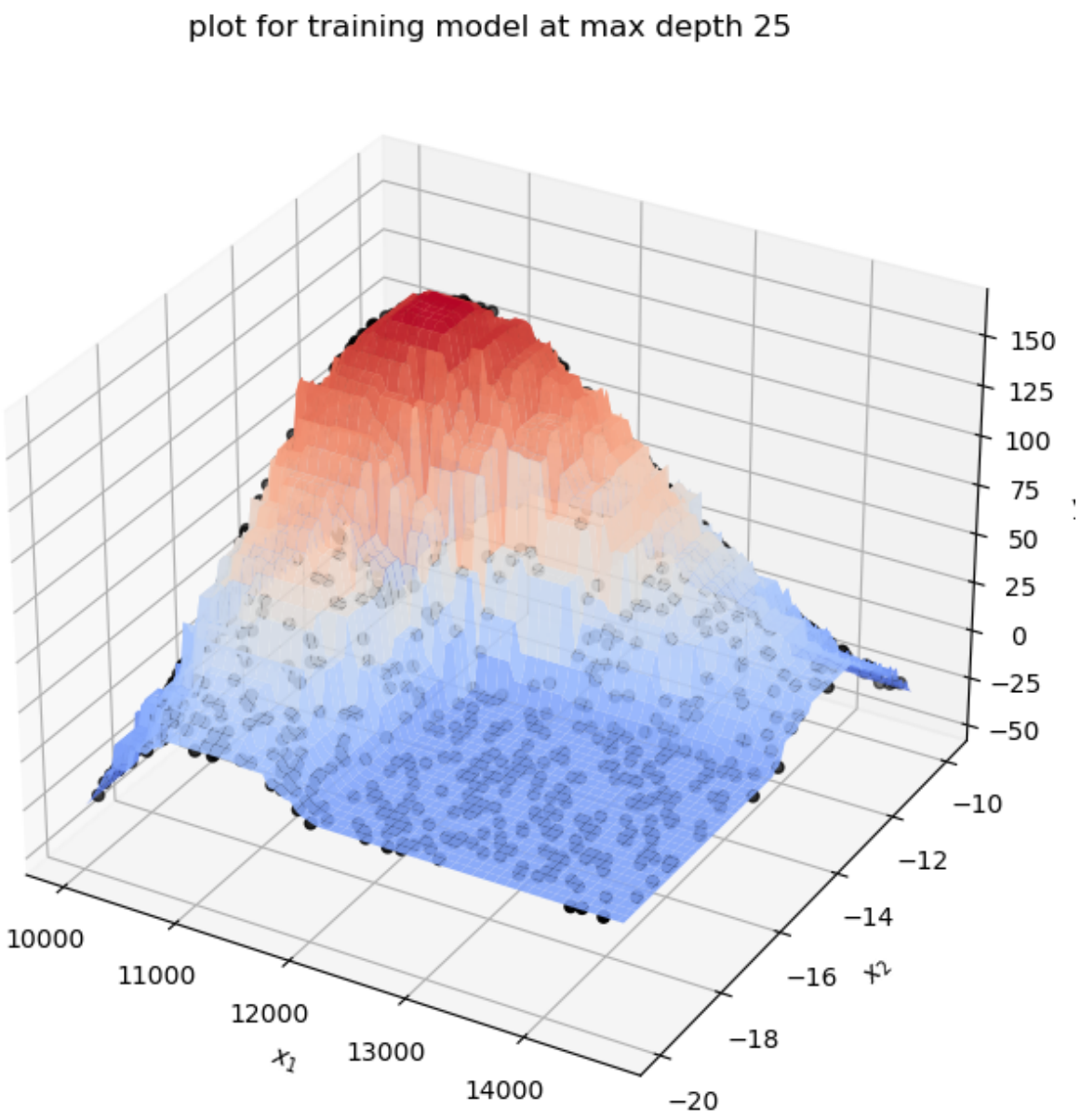


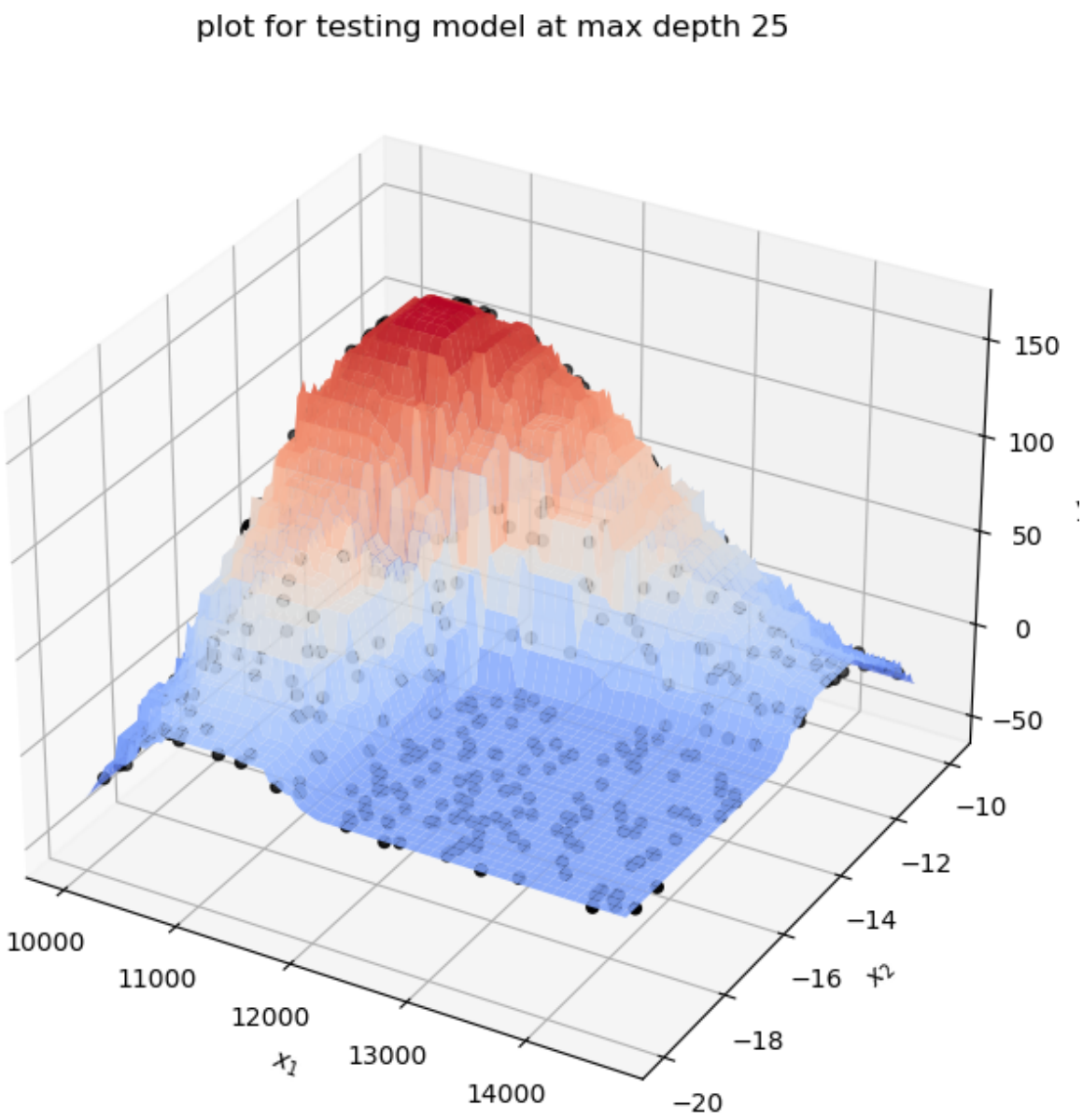

```
RMSE for train data:  4.417134916147934
RMSE for test data:  7.866244284088834
```


plot for training model at max depth 10
150
125
100
75
50
25
0
−25
−50
10000
11000
12000
13000
14000
x_1
−10
−12
−14
−16
−18
−20
x_2




```
RMSE for train data:  0.0
RMSE for test data:  6.403448273499582
```


plot for training model at max depth 25
150
125
100
75
50
25
0
−25
−50
10000
11000
12000
13000
14000
x_1
−10
−12
−14
−16
−18
−20
x_2



Question

- Which of your regression trees performed the best on testing data?

max depth of 25 gives the best regression trees as the rmse decreased to 5%

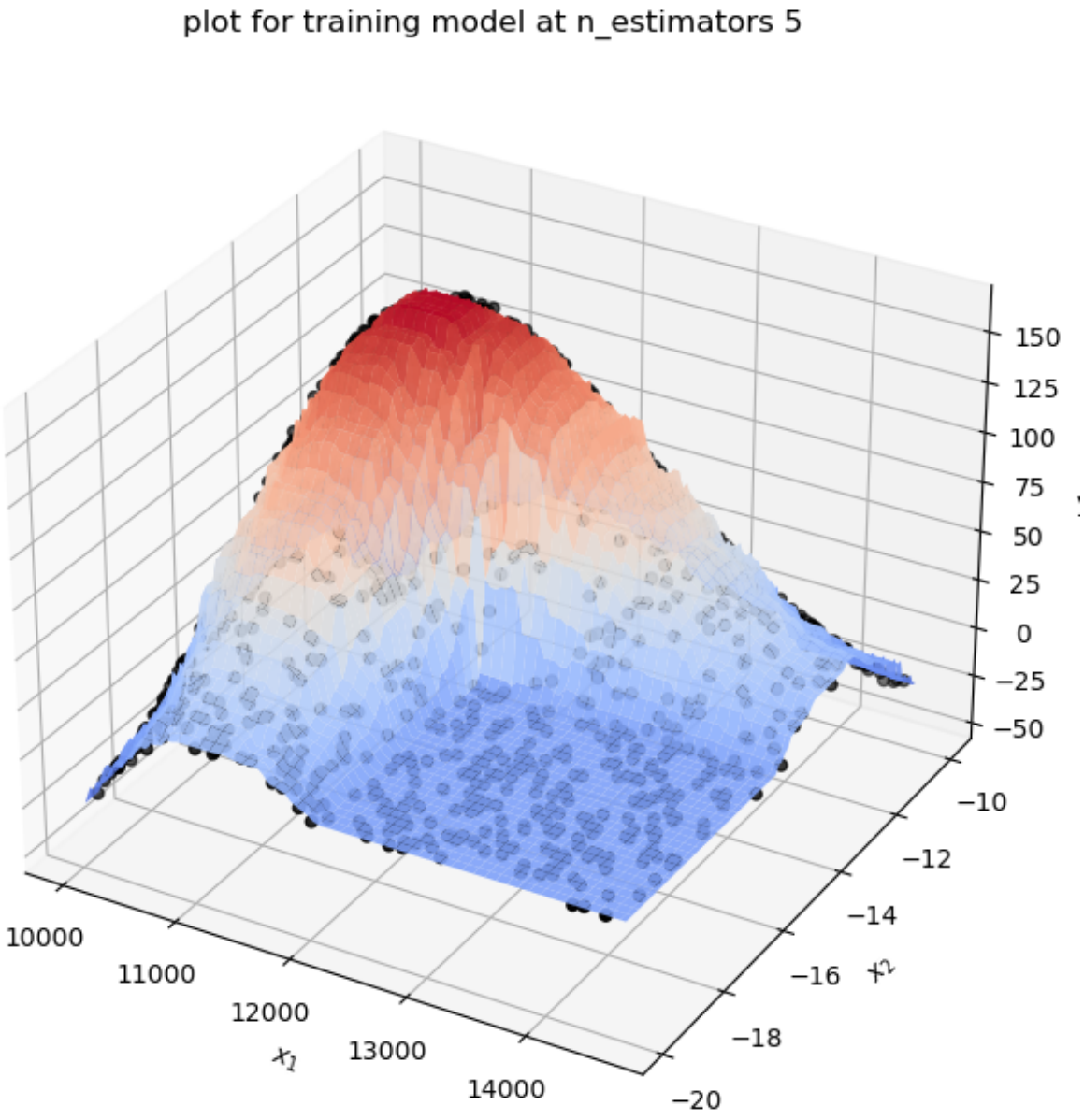
Regression trees

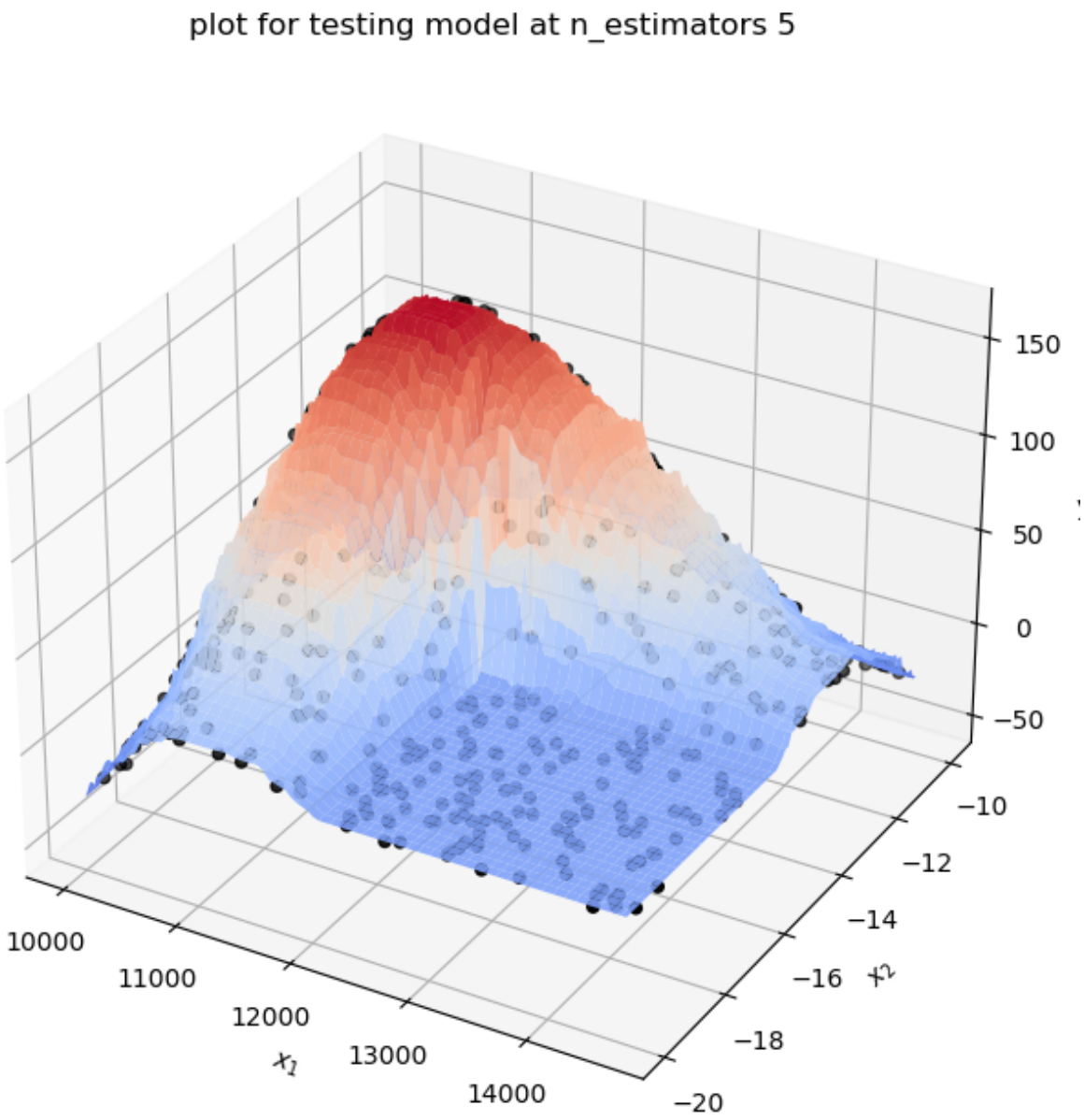
Train 4 random forests in sklearn. For all of them, use the max depth values from your best-performing regression tree. The number of estimators should vary, with values [5, 10, 25, 100].

Plot the predictions as a surface plot along with test points. Once again, for each model, compute the train and test RMSE by calling your RMSE function. Print these results.

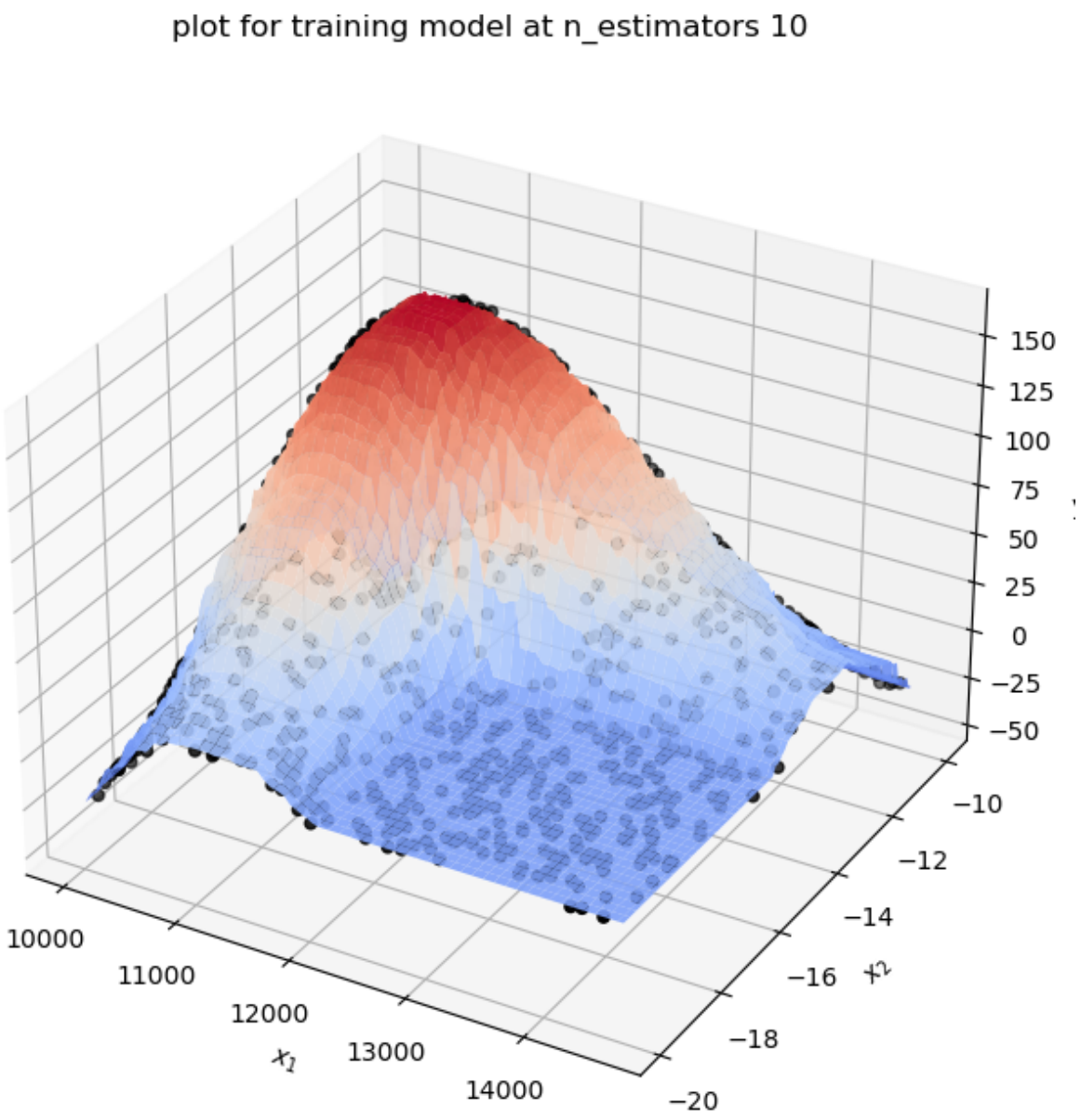
In [30]:
```
# YOUR CODE GOES HERE
for n_estimators in [5,10,25,100]:
    dt = RandomForestRegressor(max_depth = 25, n_estimators = n_estimators,random_stat
    dt.fit(train_x,train_y)
    pred_train = dt.predict(train_x)
    print("RMSE for train data: ", RMSE(train_y,pred_train))
    pred_test = dt.predict(test_x)
    print("RMSE for test data: ", RMSE(test_y,pred_test))
    make_plot(train_x,train_y,dt,f'plot for training model at n_estimators {n_estimato
    make_plot(test_x,test_y,dt,f'plot for testing model at n_estimators {n_estimators}
```

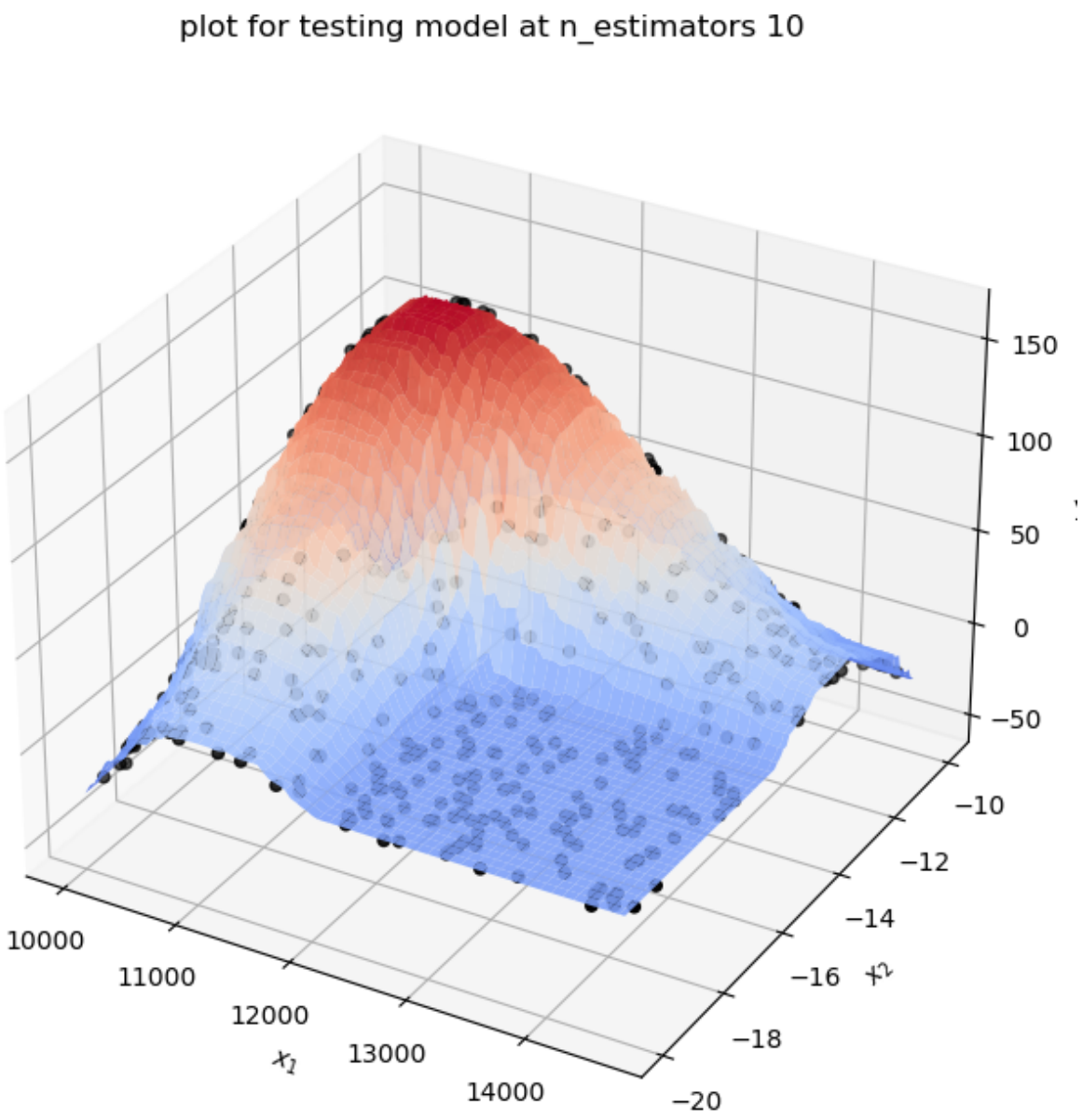
```
RMSE for train data:  2.4257706286936496
RMSE for test data:  4.821397298554881
```



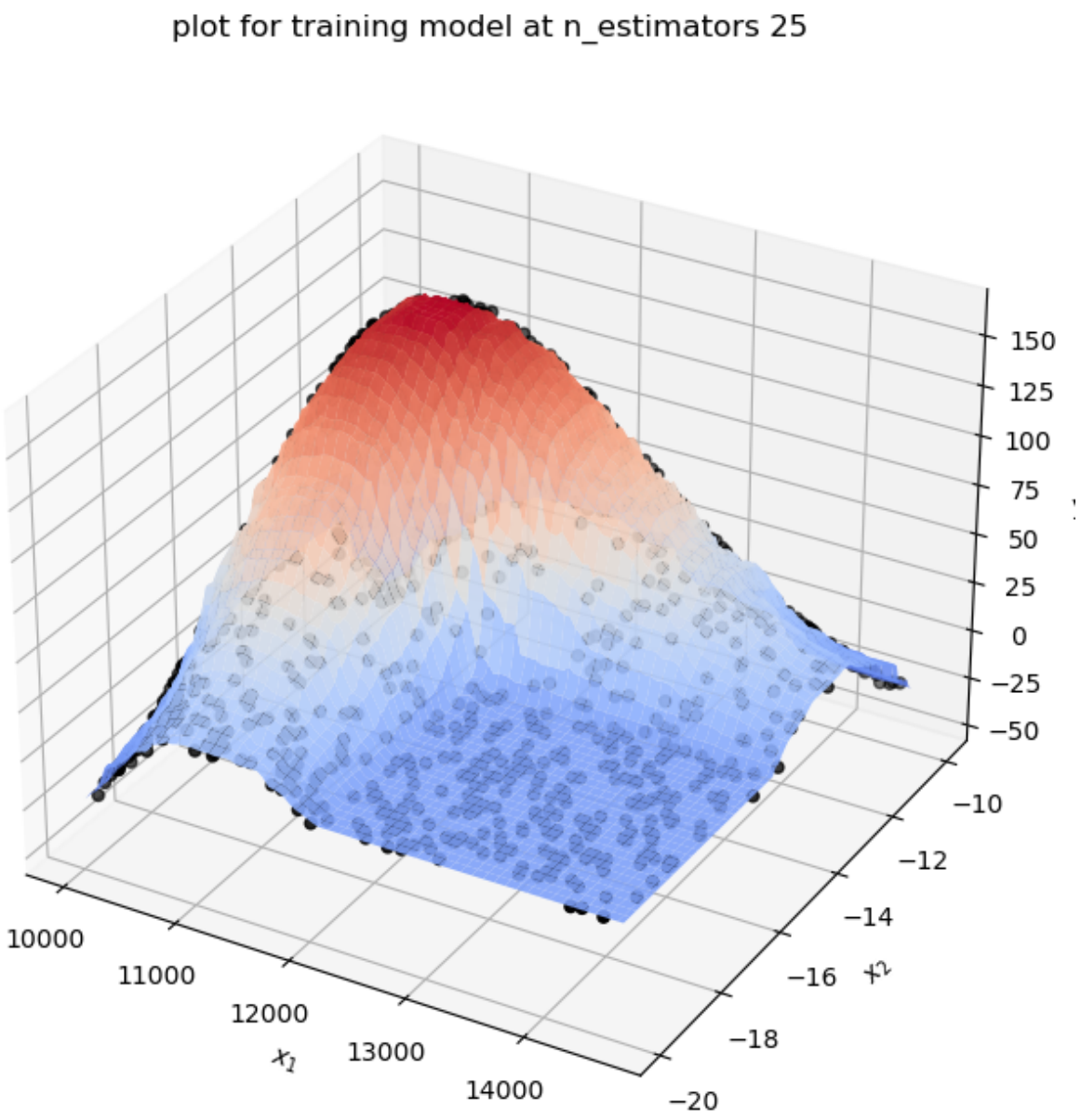


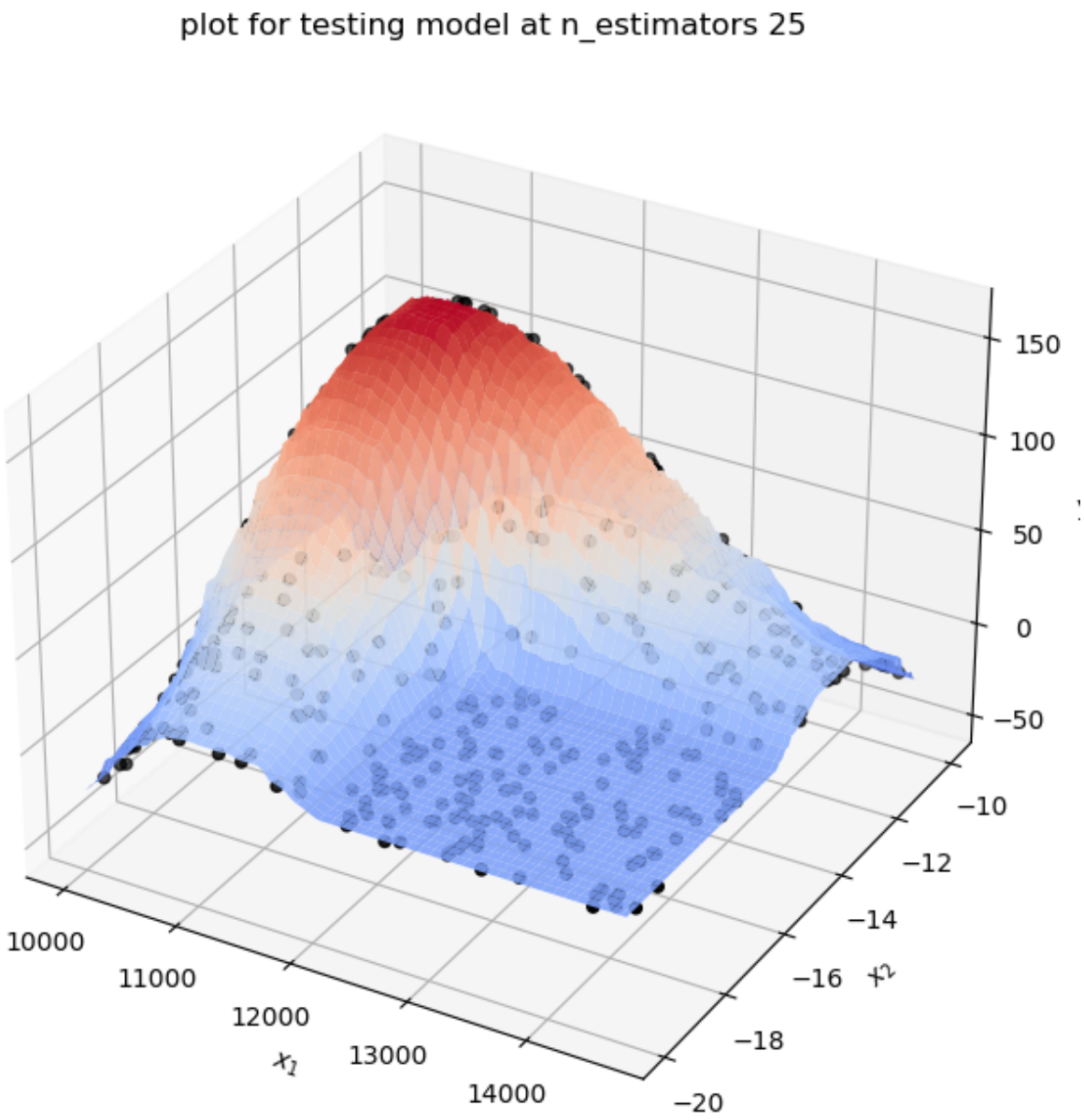

```
RMSE for train data:  1.8919014217557648
RMSE for test data:  3.93158953023002
```


plot for training model at n_estimators 10
150
125
100
75
50
25
0
−25
−50
10000
11000
12000
13000
14000
x_1
−10
−12
−14
−16
−18
−20
x_2

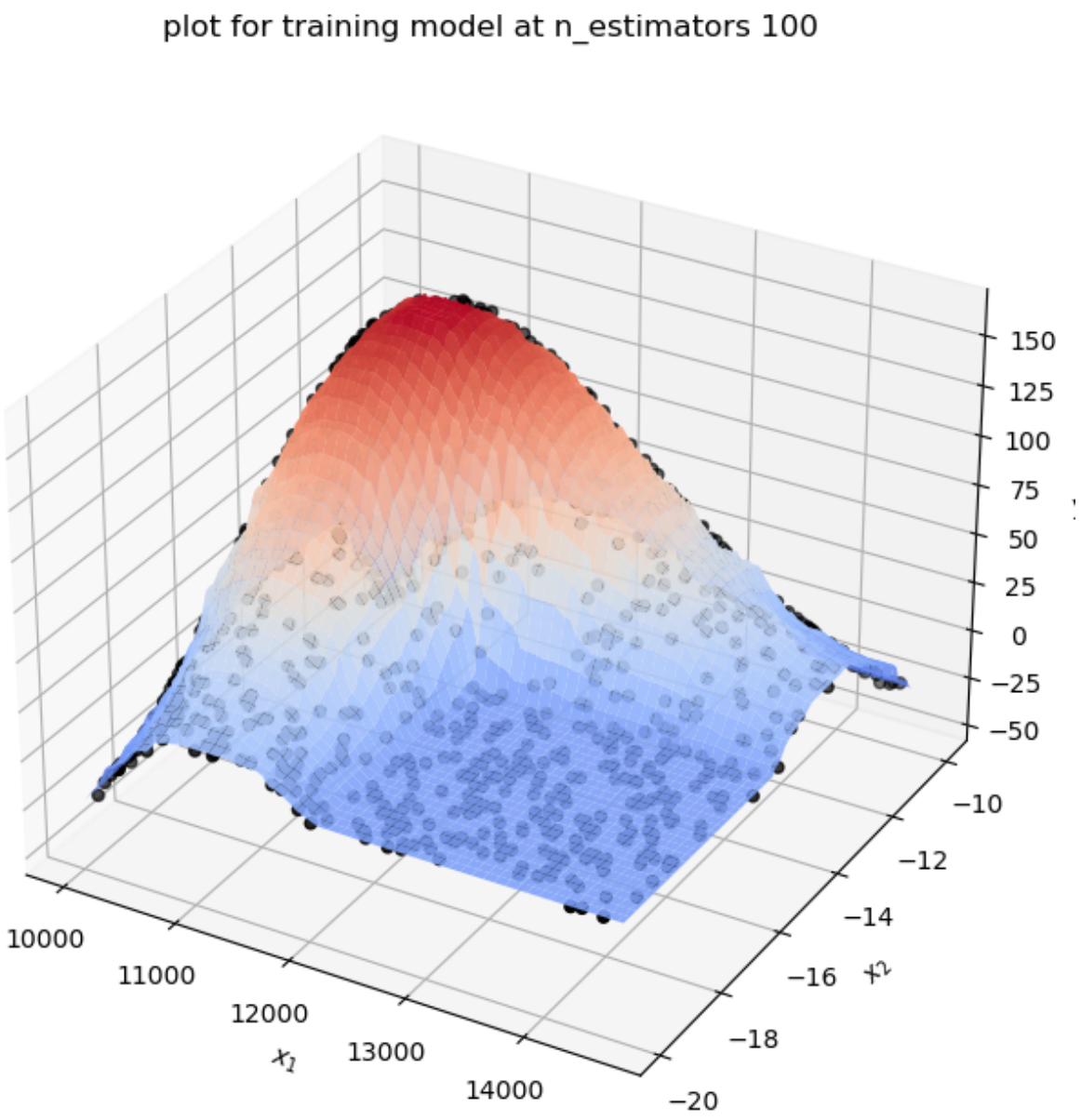


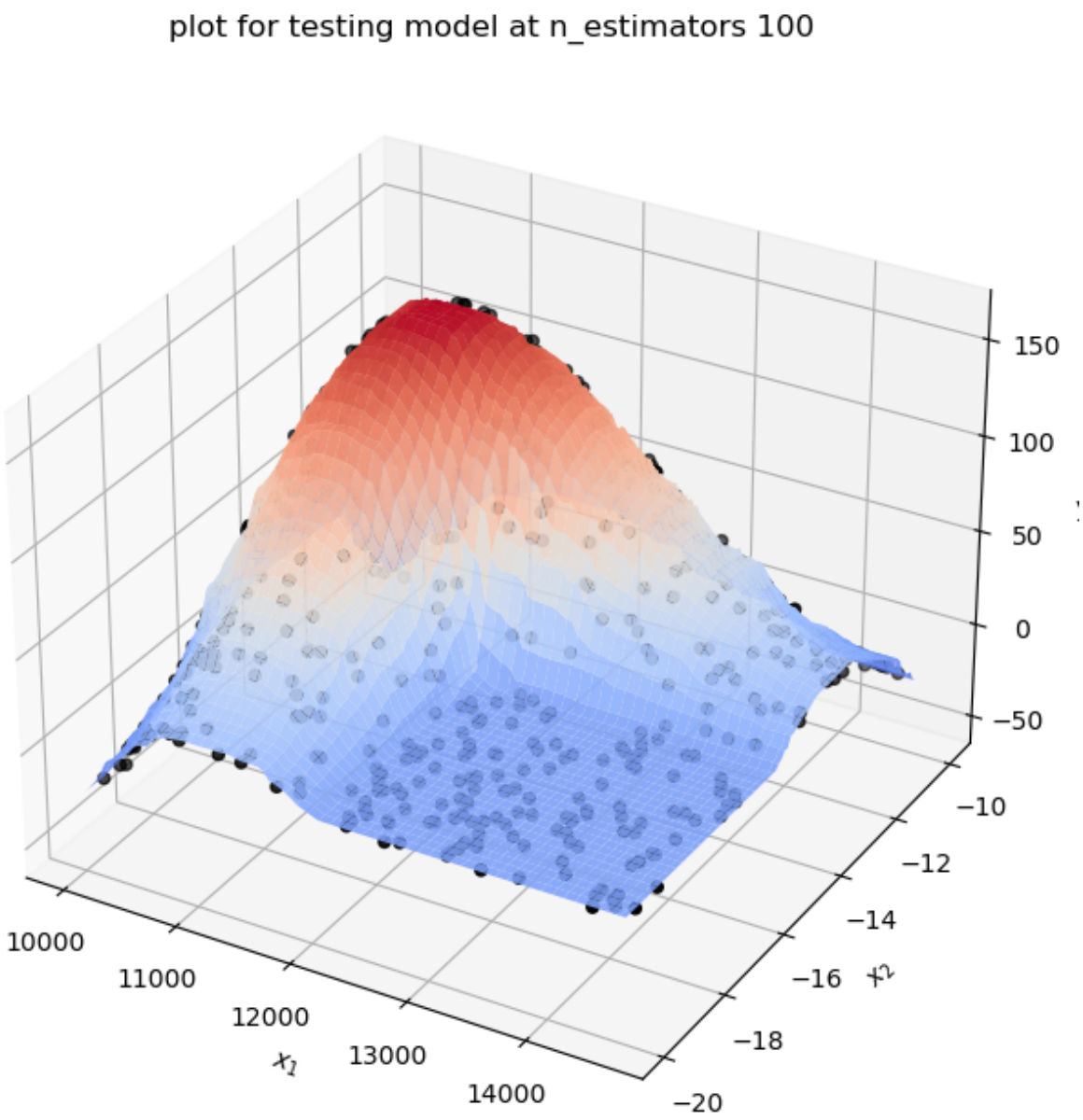

```
RMSE for train data:  1.561223767447251
RMSE for test data:  3.0837518355696627
```


plot for training model at n_estimators 25
150
125
100
75
50
25
0
−25
−50
10000
11000
12000
13000
14000
x_1
−10
−12
−14
−16
−18
−20
x_2




```
RMSE for train data:  1.31763220906486
RMSE for test data:  2.9242540919378075
```


plot for training model at n_estimators 100
150
125
100
75
50
25
0
−25
−50
10000
11000
12000
13000
14000
x_1
−10
−12
−14
−16
−18
−20
x_2



Questions

- Which of your random forests performed the best on testing data?

The n_estimators at 100 gives the best random forests model when the max depth was of 25.

- How does the random forest prediction surface differ qualitatively from that of the decision tree?

The random forest combines multiple decision trees which gives the smoother curve which is less boxy and becomes less prone to overfitting. The decision tree is less accurate on the training as compared to random forest prediction. Along with that the generalization of data is much more better in random forrest making it a better option to use.

In [ ]: